Algorytm Fortune'a – dokumentacja

Struktury danych:

Klasa Point:

Klasa point reprezentuje punkt na płaszczyźnie dwuwymiarowej za pomocą współrzędnych x i y. Jednocześnie reprezentuje zdarzenia przechowywane w strukturze zdarzeń. Do tego celu potrzebne są pola:

- orderingY - reprezentujące współrzędną sortującą zdarzenia,
- arc – parabola, którą zaczyna dany punkt
- edge – krawędź wieloboku Voronoi najbliżej danego punktu.

Metody dostępne w tej klasie to:

- __init__ - Metoda inicjalizująca punkt,
- setOrdering - Metoda ustawiająca współrzędną sortującą zdarzenia,
- toPQ - Metoda zwracająca krotkę używaną do przetrzymywania w strukturze zdarzeń,
- __hash__ - Metoda haszująca,
- __eq__ - Metoda sprawdzająca identyczność,
- __repr __ - Metoda zwracająca napis składający się z współrzędnych punktu.

Klasa HalfEdge:

Klasa HalfEdge reprezentuje półprostą lub odcinek na płaszczyźnie. Zawiera dwa pola: start, end reprezentujące początek i koniec odcinka, jeśli jedno z tych pól jest puste to mamy do czynienia z półprostą. Dodatkowymi polami są next oraz prev, dzięki nim możliwe jest utworzenie podwójnie łączonej listy krwędzi.

Metody dostępne w tej klasie to:

- __init__ - Metoda inicjalizująca (początek i koniec na tym etapie nie istnieją),
- __hash__ - Metoda haszująca,
- __eq__ - Metoda sprawdzająca identyczność,
- __repr__ - Metoda zwracająca napis składający się z punktu początkowego i końcowego.

Klasa RBNode:

Klasa RBNode reprezentuje węzeł w drzewie czerwono – czarnym, jest elementem struktury stanu. Jest używana do przedstawienia paraboli. Zawiera pola potrzebne do funkcjonowania drzewa czerwono czarnego: ojciec, lewy, prawy syn oraz kolor (parent, left, right, color). Pola reprezentujące parabolę to:

- point – punkt, który inicjuje parabolę,
- leftHalfEdge, rightHalfEdge – półproste przecinające parabolę,
- prev, next – poprzednia oraz następna parabola,
- triggeredBy – zdarzenie kołowe, które było przyczyną powstania paraboli.

Struktura stanu:

Struktura stanu reprezentowana jest przez drzewo czerwono czarne, w klasie RBTree.

Klasa oferuje standardowe metody służące do używania drzewa czerwono czarnego, takie jak: sprawdzenie czy drzewo jest puste (isEmpty), ustawienie korzenia (createRoot), lewy obrót (left_rotate), prawy obrót(right_rotate), naprawę drzewa po dodaniu elementu (fix_insert), zamianę węzłów miejscami (transplant), usunięcie węzła (delete), naprawę drzewa po usunięciu (delete_fixup), znalezienie minimalnego węzła (minimum).

Metody specyficzne da struktury stanu:

- getNodeAbove – metoda zwracająca parabolę nad danym punktem,
- insertBefore – metoda wstawiająca parabolę przed daną parabolą,
- insertAfter – metoda wstawiająca parabolę po danej paraboli,
- replace – metoda podmieniająca starą parabolę na nową (w tym samym miejscu w drzewie)

Struktura zdarzeń:

Struktura zdarzeń reprezentowana jest przez kolejkę priorytetową. Priorytet zdarzenia (punktu) jest definiowany przez pole orderingY.

Dodatkowe metody potrzebne do poprawnego działania algorytmu to:

- getIntersectionOfParabolas – zwracająca punkt przecięcia się paraboli powstałych z dwóch podanych punktów, na danej współrzędnej y,
- getConvergencePoint – zwracająca środek okręgu oraz dolny punkt okręgu powstałego z trzech podanych punktów

Klasa Voronoi:

Klasa Voronoi przechowuje oraz wyznacza diagram Voronoi dla podanego zbioru punktów.

Pola zawierające się w tej klasie to:

- points – zbiór punktów wejściowych,
- events – struktura zdarzeń,
- beachLine – struktura stanu,
- notValidEvents – zbiór przetrzymujący nieprawidłowe zdarzenia kołowe,
- vertices – zbiór punktów powstałego diagramu Voronoi,
- listEdges – lista krawędzi diagramu Voronoi,
- lowerLeft – lewy dolny punkt obramowania,
- upperRight – prawy górny punkt obramowania.

Metody wyznaczające diagram Voronoi opierają się na algorytmie opisanym w książce Marka de Berga – „Computational Geometry Algorithms and Applications”.

Główną metodą w tej klasie jest metoda „solve”, która po wywołaniu generuje diagram Voronoi.

Główny podział metod to rozróżnienie na obsługujące zdarzenia kołowe oraz zdarzenia punktowe.

Zdarzenia punktowe obsługiwane są przez metodę handleSiteEvent, wywołuje ona metody podrzędne:

- breakArc – metoda dzieląca daną parabolę na trzy na danej współrzędnej y,
- addCircleEvent – metoda dodająca zdarzenie kołowe powstałe z trzech parabol.

Zdarzenia kołowe obsługuje metoda handleCircleEvent, korzysta ona z następujących metod:

- removeArc – usuwa daną parabolę z struktury stanu,
    - addEdge – dodaje krawędź do listy krawędzi diagramu,
- addCircleEvent (opisana wyżej)

Metody pomocnicze:

- findBounds – metoda znajdująca punkty obramowania,
- getIntersectionWithBox – metoda znajdująca punkt przecięcia półprostej z obramowaniem,
- endHalfEdges – metoda kończąca wszystkie półproste (kończy w punkcie przecięcia z obramowaniem).

Wyniki działania algorytmu reprezentowane są w klasie Voronoi przez:

- pole vertices – zbiór wierzchołków diagramu,
- pole listEdges – lista krawędzi

Dodatkowo, dla każdego punktu ze zbioru wejściowego pole *edge* wskazuje na krawędź wieloboku otaczającego dany punkt, następnie dzięki polom next oraz prev krawędzi mamy dostęp do podwójnie łączonej listy krawędzi danego wieloboku.

Wizualizacja

Wizualizacja przeprowadzona za pomocą narzędzia Jupyter Notebook.

Klasa Visualization:

Klasa Visualization zajmuje się zapisywaniem do scen poszczególnych etapów działania algorytmu. Zawiera pola i metody odpowiedzialne za rysowanie poszczególnych elementów

diagramu. Dla użytkownika najważniejszym polem jest pole *scenes*, zawierające wszystkie sceny wizualizacji.

Klasa VoronoiVisualization to klasa dziedzicząca z klasy Voronoi, nadpisuje część metod w celu dodania do nich funkcjonalności związanych z wizualizacją, nowa metoda to addVisualization dodająca obiekt klasy Visualization.

Aby poprawnie przeprowadzić wizualizację należy:

1. Zdefiniować zbiór obiektów klasy Point (set),
2. Stworzyć nowy obiekt klasy VoronoiVisualization, należy przekazać do konstruktora zbiór punktów oraz ustawić flagę steps na wartość True,
3. Stworzyć nowy obiekt klasy Visualization, do konstruktora przekazujemy wcześniej utworzony obiekt Voronoi,
4. Do obiektu Voronoi dodajemy wizualizację za pomocą metody addVisualization,
5. Wykonujemy metodę solve.
6. Definiujemy nowy obiekt klasy Plot, w konstruktorze przypisujemy do scenes pole scenes obiektu Visualization.
7. Na obiekcie plot wykonujemy metodę draw, z argumentem False

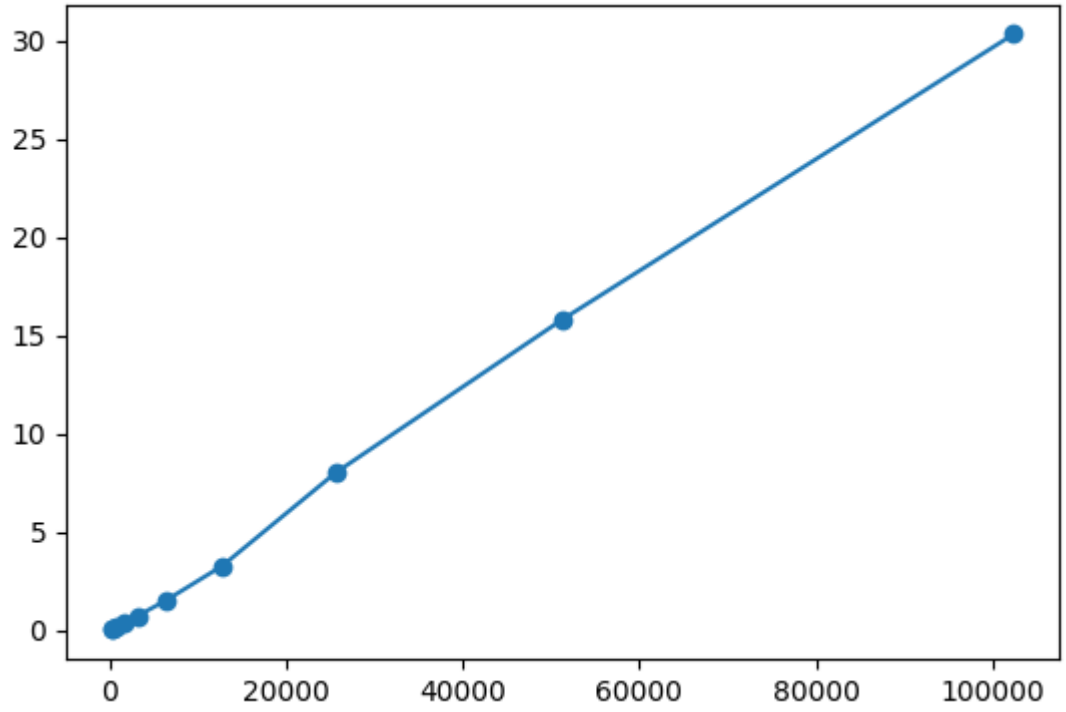
Testy i wyniki

Algorytm działa poprawnie dla zbiorów danych większych od trzech punktów, jednocześnie nieposiadających punktów mających te same współrzędne x lub y.

Testy wydajnościowe zostały przeprowadzone na losowych zestawach danych, dla każdej ilości punktów zostały powtórzone osiem razy, aby uniknąć wpływu skrajnych przypadków na czas działania. Wyniki zostały przedstawione w Tabeli 1 oraz graficznie na Rysunku 1.

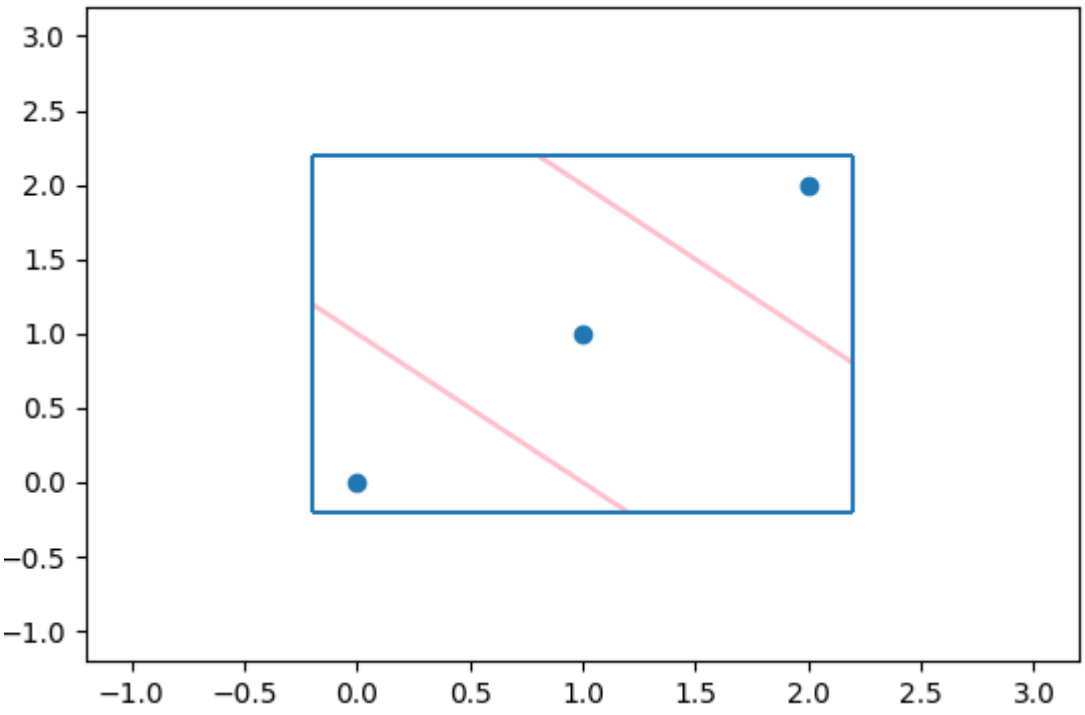
Tabela 1 Zależność czasu wykonania od ilości punktów

Liczba punktów	200	400	800	1600	3200	6400	12800	25600	51200	102400
Czas [s]	0.055	0.068	0.143	0.329	0.730	1.514	3.283	8.014	15.799	30.361

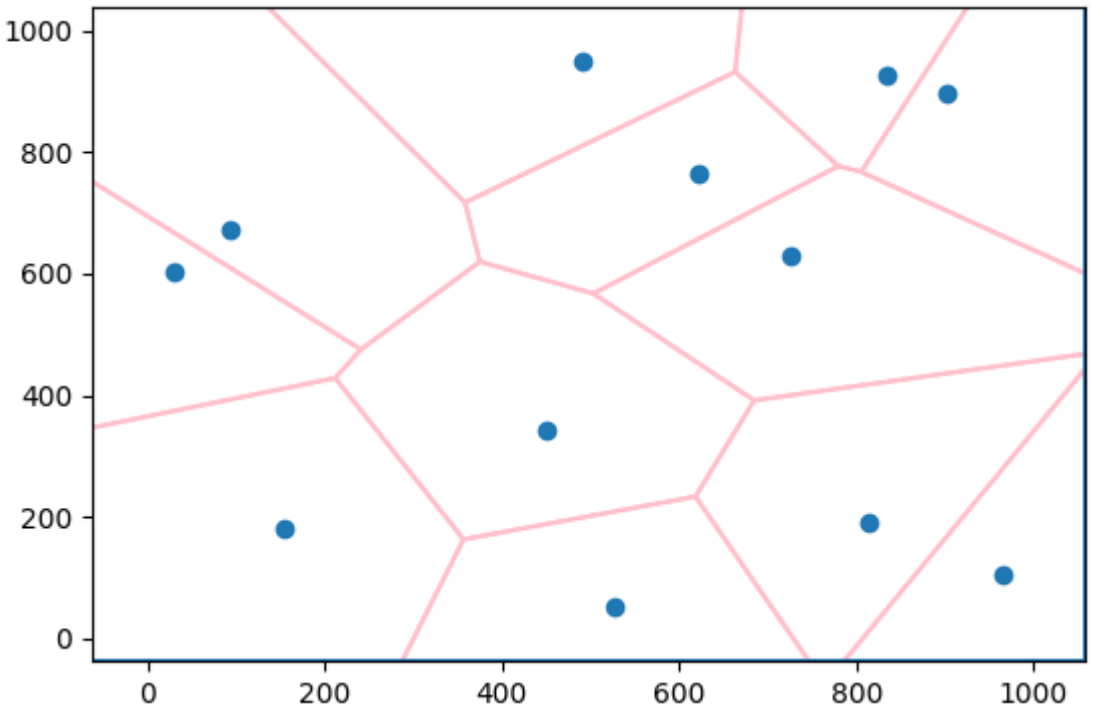


Rysunek 1 Wykres zależności czasu wykonania [s] (oś pionowa) od ilości punktów (os pozioma)

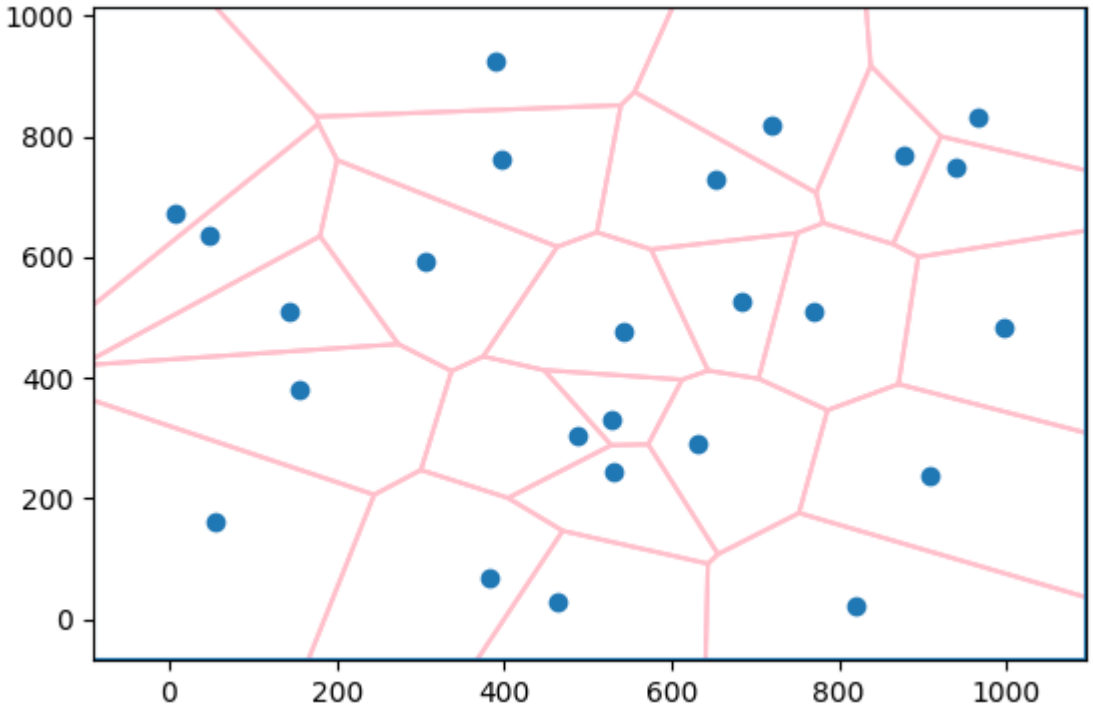
Przykładowe wyniki działania



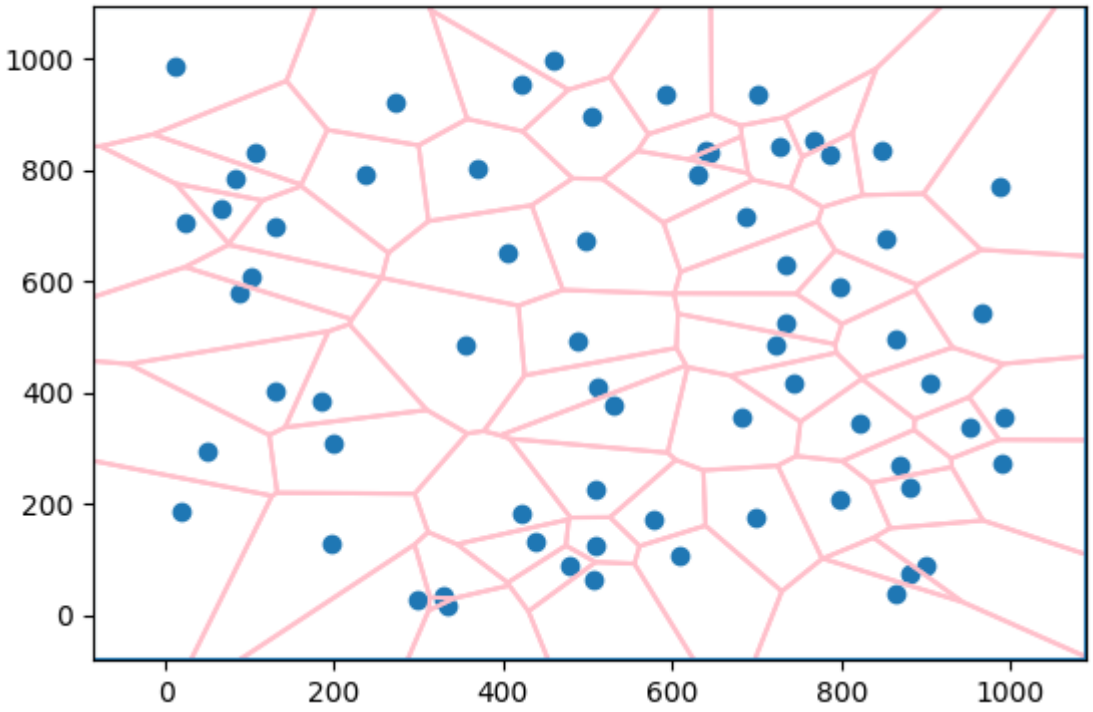
Rysunek 2 Zestaw 3 punktów leżących na prostej



Rysunek 3 Zestaw 8 losowych punktów



Rysunek 4 Zestaw 25 losowych punktów



Rysunek 5 Zestaw 70 losowych punktów

Opis algorytmu:

Algorytm Fortune'a opiera się na algorytmie zamiatania, działa w czasie O(n log n), oraz używa O(n) pamięci. Został zaprezentowany przez Stevena Fortun'ę (CZY TAK TO SIĘ PISZE WGL???) w 1986 roku.

Algorytm działa w sposób przedstawiony w książce Marka de Berga – „Computational Geometry Algorithms and Applications".

Miotłą w algorytmie jest pozioma linia poruszająca się w kierunku pionowym, z góry na dół.

Struktura stanu nazywana jest *linią brzegową*. Składa się z łuków paraboli, z których każda jest wyznaczana przez konkretny punkt zbioru wejściowego.

Punkty powyżej miotły zostały dołączone już do diagramu Voronoi, natomiast punkty poniżej nie zostały jeszcze rozważone.

Reszta opisu na prezentacji.

Linki:

Microsoft Word - cwiczenie.6b.diagramy voronoia.20090514 .doc (multimedia.edu.pl)

https://pvigier.github.io/2018/11/18/fortune-algorithm-details.html

https://math.stackexchange.com/questions/213658/get-the-equation-of-a-circle-when-given-3-points

Fortune's algorithm - Wikipedia